C#

اساسيات الدوال في لغة C# باستخدام تطبيقات الكونسل

C# Methods Foundamental Using Console Appliction

C#
Methods

إعداد: سالم مسعود الدروقي

قسم الحاسوب كلية التربية – جامعة المرقب

الدوال Methods:

هي عبارة عن مجموعة من التعليمات المجمعة مع بعضها البعض تحت اسم واحد ويمكن استخدامها اكثر من مره لأداء وظيفة معينة ولا يتم تنفيذها إلا بعد استدعائها من خلال كتابة اسمها في الدالة الرئيسية وبعد تنفيذها يتم الرجوع إلى نقطة الاستدعاء في الدالة الرئيسية ويجب ان تنتهي الدالة بعلامتي الاقواس () اثناء تعريفها واثناء استدعائها.

تساعد الدوال في تنظيم وتنسيق هيكلية البرنامج من خلال تقسيم البرنامج الرئيسي إلى مجموعة برامج فرعية (دوال) بحيث يكون لكل منها وظيفة محددة، كما تساعد في التقليل من تكرار كتابة الاسطر البرمجية (في حالة الرغبة في تنفيذها أكثر من مره) من خلال كتابتها مرة واحدة داخل دالة وإعادة تنفيذها اكثر من مرة (دون إعادة كتابتها) من خلال اعادة كتابة اسمها فقط، ويوجد في لغة C# نوعان رئيسيان من الدوال هما:

- **الدوال الجاهزة (Built-in) المبنية داخل المترجم الخاص بلغة "C#".**
- **والدوال التي يمكن تعريفها عن طريق المبرمج.**

اولاً: الدوال الجاهزة في لغة C#:

هي عبارة عن مجموعة من الدوال التي يتم برمجتها من قبل الشركة المبرمجة للغة البرمجة (C#) داخل فئات (Classes) خاصة بالنظام لكي تقوم بأداء وظيفة معينة حيث يتم برمجتها وتجهيزها لكي يتم استدعائها من قبل المبرمج لتعطي النتائج المطلوبة منها وينقسم هذا النوع من الدوال (الدوال الجاهزة) إلى نوعين:

دوال تحتاج إلى تمرير (ارسال) معاملات (ثوابت او متغيرات) حيث يتم كتابة قيم هذه المعاملات كقيم ثابتة او متغيرات تحمل قيماً بين قوسين امام اسم الدالة اثناء استدعاء الدالة.

دوال لا تحتاج إلى تمرير (إرسال) معاملات (ثوابت او متغيرات): حيث لا يتم كتابة أي قيم بين قوسي الدالة.

ومن امثلة الدوال الجاهزة دالة حساب الجذر التربيعي لعدد ما Sqrt() ودالة حساب الاس لعدد ما Pow() والجدول التالي يوضع مجموعة من اهم الدوال الجاهزة في لغة C# واستخدامها.

م	اسم الدالة	الوظيفة
1	`Abs(-x));`	إيجاد القيمة المطلقة
2	`Ceiling(x));`	التقريب إلى اقل عدد صحيح ليس اقل من x
3	`Pow(x,y));`	X مرفوعة للاس Y
4	`Sqrt(x));`	إيجاد الجدر التربيعي لـــ x
5	`Floor (x));`	التقريب إلى أكبر عدد صحيح ليس أكبر من x
6	`Max(x,y));`	إيجاد أكبر قيمة بين X,Y
7	`Min(x,y));`	إيجاد اقل قيمة بين X,Y
8	`Round (x));`	التقريب لأقرب عدد صحيح
9	`ReadLine()`	إدخال قيمة من لوحة المفاتيح
10	`WriteLine(x)`	طباعة قيمة

استدعاء الدوال الجاهزة :

كما ذكرنا سابقا فإن الدوال عبارة عن شفرة برمجية لا يتم تنفيذها إلا بعد استدعائها في البرنامج ولا استدعاء الدوال يتم كتابة اسمها مسبوقاً باسم الفئة(class) الموجودة فيها وتفصل بينهما نقطة ثم تأتي بعدها قائمة المعاملات بين قوسين (إن وجدت) والشكل التالي يمثل الشكل العام لاستدعاء دالة من الدوال الجاهزة في لغة #C.

```
Class_Name.method_Name(Argument List)
```

مثال1 : استدعاء دالة الجدر التربيعي:

```
Console.WriteLine(Math.Sqrt(25));
```

حيث إن:

Math : يمثل اسم الفئة (Class) الموجودة بداخلها الدالة.

Sqrt : يمثل اسم الدالة.

العدد 25 : يمثل المعامل الذي يتم تمريره للدالة.

ويمكن استدعاء الدالة بالطريقة التالية:

```
int y = 25;
double x = Math.Sqrt(y);
Console.WriteLine(x);
```

مع ملاحظة أن المتغير "y" في جملة الاستدعاء السابقة يمكن أن يأخذ قيمة يتم إدخالها عن طريق لوحة المفاتيح باستخدام جملة الإدخال.

مثال 2: استدعاء دالة لحساب قيمة الأس:

```
double z = Math.Pow(x, n);
```

حيث إن :

z : يمثل المتغير الذي سيستقبل القيمة المرجعة من الدالة.

Math : يمثل اسم الفئة المحتوية على الدالة.

Pow : يمثل اسم الدالة.

x,n : تمثل معاملات الدالة حيث "x" الاساس و " n " الاس.

مثال : اكتب برنامج يقوم بقراءة عدد ما ثم يقوم بطباعة الجدر التربيعي له وكذلك يقوم بإيجاد مكعب هذا العدد.

```
class Program
    {
        static void Main(string[] args)
        {
             int x ;
            double c,t;
            Console.WriteLine("Inter number");
            x=int.Parse (Console.ReadLine ());
            t=Math.Sqrt(x);
            c=Math.Pow(x, 3);
            Console.WriteLine (t);
            Console.WriteLine(c);
            Console.ReadKey();
        }
    }
```

ثانياً : الدوال المعرفة من قبل المبرمج:

وهي مجموعة الدوال التي يتم إنشائها من قبل المبرمج لأداء وظيفة معينة وتتكون هذه الدوال جزئيين رئيسيين هما رأس الدالة (Method Header) و وجسم الدالة (Method Body)

```
<Method Header>()
{
<Method Body >
}
```

ويمكن أن يتم تصنيف الدوال المعرفة من قبل المبرمج حسب اربعة معايير رئيسية:

أولاً: حسب قابلية الوصول Access Modifier: تنقسم الدوال والمتغيرات من حيث امكانية الوصول إليها إلى:

- **عامة public:** يمكن الوصول إليها من كافة الفئات في المشروع.
- **خاصة private:** لا يمكن الوصول إليها إلا من داخل الفئة المعرفة فيها.
- **محمية protected:** لا يمكن الوصول إليها إلا من خلال من الفئة المعرفة فيها والفئات الموروثة منها.
- **داخلية Internal:** لا يمكن الوصول إليها إلا من خلال الفئة المعرفة فيها والفئات الموجودة معها في نفس الاسمبلي (**Assembly**).
- **داخلية محمية Protected Internal:** لا يمكن الوصول إليها إلا من خلال الفئة المعرفة معها في نفس الاسمبلي (**Assembly**) والفئات المشتقة منها الموجودة في اسمبلي اخر.

ثانياً: حسب المشاركة بين الكائنات Static or Non Static : عند اشتقاق (Objects) مجموعة من الكائنات من نفس الفئة (Class) فإن أي من هذه الكائنات له قيم الخاصة به في كافة الطرق

(الدوال) والمتغيرات وكل منها في موقع مختلف في الذاكرة وفي حالة الرغبة في جعل متغير أو دالة مشتركة (موقع واحد في الذاكرة لجميع الكائنات) بين كافة الكائنات فإنه يجب تعريفها على اساس أنها ساكنه "static" وبهذا تم تصنيف الدوال حسب مشاركة الكائنات إلى نوعين :

- **دالة غير مشتركة بين كافة الكائنات المشتقة من الفئة non static:** أي أنه لكل كائن مشتق من الفئة قيماً خاصة لهذه الدوال والمتغيرات وفي مواقع مختلفة من الذاكرة وتسمى هذه الدالة " instance member" وعند استدعاء هذه الدالة فإنه يجب أن يتم اشتقاق كائن(object) من الفئة (Class) المحتوية على الدالة اولاً ثم يتم استدعاء الدالة خلال كتابة اسم الدالة بعد اسم الكائن المشتق من الفئة (بدلاً من كتابة اسم الفئة مباشرةً) تفصل بينهما نقطة وبالتالي فإن مجموعة المتغيرات والدوال من هذا النوع يتم تهيئتها (initialized) في كل مرة يتم فيها اشتقاق كائن من الفئة.

- **دالة مشتركة بين كافة الكائنات المشتقة من الفئة Static:** أي أن هذه الدالة لها موقع واحد في الذاكرة تشترك فيه كافة الكائنات وتسمى "Class member" ويتم إضافة الكلمة المحجوزة "static" في رأس الدالة اثناء تعريفها، وعند هذا النوع من الدوال لا تحتاج إلى اشتقاق نسخة من الفئة (الكلاس/class) بل يتم كتابة اسم الكلاس مباشرة وتليه اسم الدالة تفصل بينهما نقطة وبالتالي فإن مجموعة المتغيرات والدوال من هذا النوع لا يتم تهيئتها (initialized) إلا مرة واحدة عند تحميل الفئة فقط وليس عند اشتقاق كائن من الفئة.

ثالثاً: حسب القيمة المرجعة: تنقسم الدوال من حيث امكانية ارجاع قيمة أو لا إلى:

- **دوال يمكن أن ترجع قيمة**: حيث يقوم هذا النوع من الدوال بإرجاع قيمة (يتم تحديد نوعها اثناء تعريف الدالة) إلى سطر الاستدعاء (السطر الذي تم عنده الاستدعاء) (Calling Code) بعد انتهاء التنفيذ.
- **دوال لا ترجع قيمة**: وهي دوال تقوم بتنفيذ تعليمات معينة دون أن تقوم بإرجاع قيمة إلى سطر الاستدعاء (السطر الذي تم عنده الاستدعاء) (Calling code) بعد انتهاء التنفيذ ويتم تحديد هذا النوع من الدوال اثناء تعريف الدالة باستخدام الكلمة المحجوزة "void" في راس الدالة.

رابعاً: حسب المعاملات Parameters: يمكن تصنيف الدوال من حيث وجود المعاملات إلى:

- **دوال تحتوي على معاملات**: وهي دوال تحتاج إلى تمرير (ارسال) معاملات (ثوابت أو متغيرات) اثناء استدعائها، حيث يتم كتابة قيم هذه المعاملات كقيم ثابتة أو متغيرات تحمل قيماً بين القوسين () امام اسم الدالة اثناء استدعاء الدالة.

- **دوال لا تحتوي على معاملات:** وهي دوال لا تحتاج إلى تمرير قيم اثناء استدعائها حيث لا يتم كتابة اي قيم بين قوسي الدالة في رأس الدالة.

وفيما يلي نقدم الشكل العام لطريقة كتابة هذه الدوال في لغة #C:

```
class Program
 {
     static void Main(string[] args)
     {

     }
     public void Method1()
     {
      // Method body

     }
     public void Method2( int x ,double y)
     {

         // Method body

     }
     public int Method3()
     {
         // Method body
         return 0;

     }
     public int Method4(int x, double y)
     {
         // Method body
         return 0;

     }

      Public Static void Method5()
     {

         // Method body

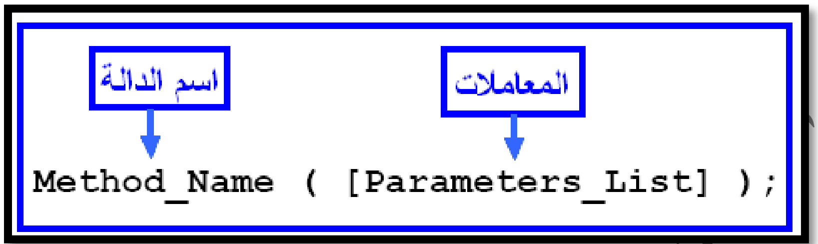
     }
 }
```

- Main: الدالة الرئيسية
- Method1: دالة لا تستقبل معاملات ولا ترجع قيمة
- Method2: دالة تستقبل معاملات و لا ترجع قيمة
- Method3: دالة لا تستقبل معاملات وترجع قيمة
- Method4: دالة تستقبل معاملات وترجع قيمة
- Method5: دالة مشتركة بين كافة الكائنات

استدعاء الدوال داخل البرنامج وتمرير المعاملات:

أولاً: استدعاء الدوال:

كما هو في الحال في الدوال الجاهزة (Built-in) فإنه يتم استدعاء الدوال المعرفة من قبل المبرمج عن طريق كتابة اسمها في البرنامج متبوعاً بقوسين يتم بينهما كتابة قيم المعاملات المطلوب تمريرها إلى الدالة إن وجدت والصيغة العامة لاستدعاء الدالة كالآتي:



ثانياً: تمرير المعاملات إلى الدالة:

كما ذكرنا سابقاً أن هنالك بعض الدوال يتم تعريفها لكي تستقبل قيمة أو مجموعة من القيم يتم تمريرها لها اثناء استدعائها من خلال كتابة قيم أو متغيرات مناظرة للمتغيرات المعرفة في راس الدالة في جملة الاستدعاء وتسمي المعاملات (المتغيرات) الموجودة في الدالة الرئيسية (المكتوبة امام جملة الاستدعاء) بالمعاملات الفعلية بينما تسمى المعاملات (المتغيرات) المناظرة لها في رأس الدالة بالمعاملات الصورية أو الشكلية مع ملاحظة أنه يجب ان تتطابق المعاملات الفعلية (المكتوبة في جملة الاستدعاء) والمعاملات الصورية (الموجودة في رأس الدالة) من حيث انواعها وعددها وترتيبها.

طرق تمرير المعاملات إلى الدالة:

توجد هنالك ثلاث طرق لعملية تمرير المعاملات إلى الدالة في لغة #C يتم استخدام إحداها حسب الحاجة:

الطريقة الأولى: تمرير المعاملات بالقيمة Pass-By-Value: وهي الطريقة الافتراضية لإرسال المعاملات في لغة #C، وفيها يتم إرسال نسخة من المعامل الفعلي إلى المعامل الصوري (الشكلي) بمعنى أن كلا المعاملين (الفعلي والشكلي) لا يشتركان في موقع واحد في الذاكرة، وبالتالي فإن أي تغير في قيمة المعامل الشكلي (داخل الدالة) لا ينتج عنها تغير في المعامل الفعلي (داخل البرنامج الرئيسي) وهذا النوع من تمرير المعاملات يتم تطبيقه افتراضياً عند تمرير المعاملات إلى الدالة.

مثال: المثال التالي يقوم باستدعاء دالة تقوم بحساب مربع عدد (x) يتم إرساله لها، ومن خلال ناتج تنفيذ البرنامج و من خلال طباعة قيمة المتغير "x" قبل إرساله إلى الدالة وطباعته بعد إرساله إلى الدالة نلاحظ أن قيمته لم تتأثر بالعملية التي حدثت داخل الدالة نظراً لّن العملية التي حدثت على المتغير داخل الدالة تم تخزينها في موقع اخر من الذاكرة بعيد عن المتغير الفعلي.

```
class Program
  {
    static void Main(string[] args)
    {
     int x = 10;
     Console.WriteLine("The value of x before change is "+ x) ;
     change(x); //  استدعاء الدالة
     Console.WriteLine("The value of x after change is " + x);
     Console.ReadKey();
     } //end of main
     static void change( int y)
     {
      y =y*y ;
     }//end of Method
    }
```


مخرجات البرنامج :

```
The value of x before change is 10
The value of x after change is 10
```

من خلال المثال السابق نلاحظ أن قيمة المتغير "x" لم تتغير قبل جملة الاستدعاء وبعد جملة الاستدعاء وذلك نظراً لاستخدام اسلوب التمرير بالقيمة.

الطريقة الثانية: تمرير المعاملات بالعنوان (المرجع) (بالإشارة): Pass-By-Reference: وفيها يتم إرسال عنوان المعامل الفعلي (الموجود في جملة الاستدعاء) في الذاكرة إلى المعامل الصوري المناظر له في الدالة وبالتالي فإنه وفي هذه الحالة فإن المعامل الفعلي و المعامل الشكلي يشيران إلى نفس موقع

الذاكرة (يشتركان في موقع الذاكرة) وينتج عن ذلك أن أي تغير يحدث في قيمة المعامل الشكلي فإنه سيتم تطبيق هذا التغير على المعامل الفعلي في الدالة الرئيسية وهذا النوع من تمرير المعاملات يحتاج عند تطبيقه الى كتابة الكلمة المحجوزة "Ref" قبل اسم المعامل الصوري والمعامل الفعلي.

مثال: سنقوم في هذا المثال بإعادة نفس المثال السابق مع تمرير قيمة المعامل "x" بالمرجع.

```
class Program
    {
     static void Main(string[] args)
     {
      int x = 10;
      Console.WriteLine("The value of x before change is"+ x) ;
      change( ref x);
      Console.WriteLine("The value of x after change is"+ x);
      Console.ReadKey();
     } //end of main

     static void change(ref int y){
         y =y*y ;
        }//end of Method
    }
```

مخرجات البرنامج :

```
The value of x before change is 10
The value of x after change is 100
```

من خلال المثال السابق نلاحظ أن قيمة المتغير "x" تغيرت بعد جملة الاستدعاء وذلك نظراً لاستخدام اسلوب التمرير بالمرجع.

مثال: برنامج يقوم بتعريف دالة تستقبل عددين وتقوم بإيجاد حاصل جمع العددين على أن تتم طباعة الناتج داخل الدالة.

```
class Program
    {
        static void Main(string[] args)
        {
            sum(10, 20);
            Console.ReadKey();
        }
        public static void sum( int x ,int y)
        {
            int c;
            c = x + y;
            Console.WriteLine("Sum=" + c);
        }

    }
```

مثال: برنامج يقوم بتعريف دالة تستقبل عددين وتقوم بإيجاد حاصل جمع العددين على أن يتم ارجاع ناتج الجمع للدالة الرئيسية.

```
class Program
    {
        static void Main(string[] args)
        {
            int A = 10, B = 20, w;
            w= sum(A, B);
            Console.WriteLine("Sum=" + w);
            Console.ReadKey();
        }
        public static int sum( int x ,int y)
        {
            int c;
            c = x + y;
            return c;
        }
    }
```

امثله:

1- اكتب برنامج يقوم باستدعاء دالة تقوم بحساب مساحة مثلث من خلال المعادلة (مساحة المثلث = نصف القاعدة* الارتفاع) علماً بأن قاعدة المثلث =تساوي 4.5 متر وارتفاعه 6.6 متر على أن يتم طباعة المساحة في الدالة.

```
class Program
    {
        static void Main(string[] args)
        {
            double B = 4.5, H = 6.6;
            triangle_area(B, H);
            Console.ReadKey();
        } //end of main

        public static void triangle_area(double x,double y)
        {
          double area;
          area=(0.5*x)*y;
          Console.WriteLine ("Area="+area);
        }//end of Method
    }
```

2- اكتب برنامج يقوم بقراءة نصف قطر دائرة ثم يقوم باستدعاء دالة تقوم بحساب مساحه الدائرة على أن تتم الطباعة في الدالة الرئيسية (main).

```
class Program
    {
      static void Main(string[] args)
       {
        double R, area;
        const  double pi=3.14;
        Console.WriteLine ("Enter R " );
        R=int.Parse (Console.ReadLine ());
        area= cir_area(R,pi);
        Console.WriteLine("Area=" + area);
        Console.ReadLine();
       }
    public static double cir_area(double Rad, double pi)
    {
        return pi*(Math.Pow(Rad,2));
     }//end of Method
    }
```

3- اكتب برنامج يقوم بقراءة عدد صحيح (n) ثم يقوم باستدعاء دالة تقوم بطباعة الاعداد من 1 إلى العدد المدخل من قبل المستخدم (n) على النحو التالي:

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
..................n

```
class Program
    {
        static void Main(string[] args)
        {
          int n;
          Console.WriteLine ("Enter Number" );
          n=int.Parse (Console.ReadLine ());
          num(n);
          Console.ReadKey();
          }
         static void num(int x)
          {
              for (int i=1;i<=x;i++)
                 {
                    for(int j = 1 ; j<=i ;j++)
                     Console.Write (j);
                    Console.WriteLine();
                 }
           }
        }
```

-4 اكتب برنامج يقوم بقراءة 10 اعدد ثم يقوم باستدعاء داله تقوم بحساب مجموع القيم لها على أن تتم الطباعة في البرنامج الرئيسي.

```
class Program
   {
        static int sum = 0;
        static void Main(string[] args)
        {
            int x ,i , sumation=0;
            for (i=1;i<=10;i++)
            {
               Console.Write("number "+i+"=");
               x=int.Parse (Console.ReadLine ());
               sumation = suma(x);
            }
         Console.WriteLine("sumation= " + sumation);
         Console.ReadKey();
        }
        public static int suma(int num)
        {
            sum = sum + num;
            return sum;
        }
   }
```

5- اكتب برنامج يقوم بتعريف دالة تقوم بقراءة بيانات موظف (الاسم ، الحالة الاجتماعية (0 او 1) ، الراتب الاساسي) وكذلك دالة اخرى تقوم بحساب صافي المرتب بحيث يتم اضافة نسبة 10 % على الراتب الاساسي إذا كان الموظف متزوج (الحالة الاجتماعية =1) و 5 % اذا كان غير متزوج (الحالة الاجتماعية =0) ودالة ثالثة تقوم بطباعة اسم الموظف و الراتب الاساسي وصافي المرتب.

```
class Program
    {
        static void Main(string[] args)
        {
            double net_in = 0;
            int stat;
            Console.Write("Enter name: ");
            String name = Console.ReadLine();
            Console.Write("Enter Salary= ");
            double sal = double.Parse(Console.ReadLine());
            Console.Write("Enter socity=");
            stat = int.Parse(Console.ReadLine());
             net_in = netmony(sal, stat);
            print(name, sal, net_in);
            Console.ReadLine();
        }
        public static double netmony(double Sa, int st)
        {
            double com = 0, net = 0;
            if (st == 1)
                com = Sa * 0.10;
            else if (st == 0)
                com = Sa * 0.05;
            net = Sa + com;
            return net;
        }
public static void print(String nam, double sal, double net_sal)
        {
            Console.WriteLine("Name:  " + nam);
            Console.WriteLine("Salary= " + sal);
            Console.WriteLine("Net Salary=" + net_sal);
        }
}
```

6- اكتب برنامج يقوم باستقبال قيمة عائد مشروع تجاري وكذلك قيمة المصروفات ثم يقوم باستدعاء دالة تقوم بتقسيم الارباح على شركاء المشروع الثلاثة بنسبة 1:2:3.

```
class Program
  {
     static void Main(string[] args)
     {
      Console.Write("Enter Total income=" );
      double income=double.Parse(Console.ReadLine ());
      Console.Write("Enter outgoing=" );
      double M=double.Parse(Console.ReadLine ());
      double  net_In=income-M;
      Console.WriteLine ("Total income = "+income);
      Console.WriteLine("Total Project Profit = " + net_In);
      Console.WriteLine("Total outgoing = " + M);
      ok(net_In);
      Console.ReadKey();
    }
    public static void ok(double net_income)
    {
      double n1,n2,n3;
      n1= net_income*1/6;
      n2= net_income*2/6;
      n3= net_income*3/6;
      Console.WriteLine("Share of the first partner = " + n1);
      Console.WriteLine("Share of the second partner = " + n2);
      Console.WriteLine("Share of the Third partner = " + n3);
    }
 }
```


7- اكتب برنامج يقوم بقراءة اسم الموظف وراتبه الاساسي ثم يقوم باستدعاء دالة تقوم بحساب صافي المرتب حيث أن قيمة الخصم تبلغ 5% من إجمالي الراتب الاساسي وفي حالة تجاوز الراتب 1000 يقوم باستدعاء دالة اخرى تقوم بحساب صافي المرتب حيث أن قيمة الخصم تبلغ 10% ثم يقوم استدعاء دالة تقوم بطباعة الاسم و صافي المرتب.

```
class Program
    {
        static void Main(string[] args)
        {

           String name;
           int sal;
           double net;
           Console.Write("Enter Name=" );
           name = Console.ReadLine ();
           Console.Write("Enter Salary=");
           sal = int.Parse (Console.ReadLine ());
           if (sal <= 1000)
           {
               net = tax5(sal);
               print(name, net);
           }
           else
           {
               net = tax10(sal);
               print(name, net);
           }
           Console.ReadKey();
        }
        public static double tax5(int num)
        {
            double tax, net_sal;
            tax = num * 0.05;
            net_sal = num - tax;
            return net_sal;
        }
        public static double tax10(int num)
        {
            double tax, net_sal;
            tax = num * 0.10;
            return net_sal = num - tax;
        }
        public static void print(String nam, double net)
{
 Console.WriteLine ("name: "+ nam);
 Console.WriteLine ("net= "+ net);
}
```

الدوال المصفوفات Methods & Arrays:

عندما درسنا موضوع المصفوفات عرفنا أن المصفوفات تستخدم للتقليل من عدد المتغيرات في البرنامج من خلال تخزين عدد من القيم المتشابه (من نوع واحد) تحت اسم واحد وبالتالي توفر امكانية إدخال عدد كبير من القيم دون الحاجة لتعريف عدد مماثل لها من المتغيرات وكما هو الحال عند استخدام الدوال وكما نلاحظ في كافة الامثلة السابقة أن عدد المعاملات التي يتم تمريرها إلى الدالة محدود بعدد معين من البارامترات وفي حالة الرغبة في إرسال (تمرير) عدد كبير من القيم فإنه يجب استخدام (تعريف) عدد مماثل (مساوي) له من المعاملات وهذا غير مجدي في حالة الرغبة في إرسال عدد كبير من القيم ومن هنا جاءت فكرة استخدام المصفوفات مع الدوال وذلك للتقليل من عدد المتغيرات المعرفة كبارامترات في الدالة من خلال تعريف دالة تستقبل عدد من القيم المجمعة تحت اسم واحد متمثل في مصفوفة من البارامترات.

الدوال والمصفوفات ذات البعد الواحد:

الشكل العام لتعريف دالة تستقبل مصفوفة من البارامترات

لا يوجد هنالك فرق كبير بين تعريف دالة تحتوي على مجموعة معاملات وبين دالة تحتوي على مصفوفة من المعاملات حيث إن الفرق الوحيد يكمن في تغير قائمة المعاملات بمصفوفة من المعاملات اثناء التعريف.

- في حالة استقبال مصفوفة وارجاع قيمة:

```
Access_modifier   returen_type   method_name (array_type []   param_array_name)
{
}
```

- في حالة استقبال مصفوفة و إرجاع مصفوفة:

```
Access_modifier   returen_type[]    method_name (array_type []    param_array_name)
{
}
```

مثال :اكتب شفرة الإعلان عن دالة تقوم باستقبال مصفوفة مكونه من 5 قيم ثم تقوم بطباعة هذه القيم (لا تعود بأي قيمة).

```
public static void print_arr( int[] arr)
    {
        for (int i =0 ;i <5;i++)
            Console.WriteLine(arr[i]+" ");
    }
```

الشكل العام لاستدعاء دالة و إرسال مصفوفة من البارامترات لها:

في حالة كانت الدالة لا ترجع أي قيمة   method_name(array_name)

في حالة كانت الدالة ترجع قيمة   variable_name= method_name(array_name)

مثال : اكتب برنامج يقوم بتخزين قيم لعناصر مصفوفة ثم يقوم باستدعاء دالة تقوم بطباعة هذه القيم.

```
class Program
    {
        static void Main(string[] args)
        {
            int[] m= new int[5];
          m[0]=5; m[1]=15; m[2]=55;
          m[3]=3; m[4]=7;
          print_arr(m);
          Console.ReadKey();
    }
    public static void print_arr(int[] arr)
    {
        for (int i =0 ;i <5;i++)
           Console.WriteLine (arr[i]+" ");

    }

  }
```

مثال 1 : اكتب برنامج يقوم باستدعاء دالة تقوم بقراءة عناصر مصفوفة مكونة 5 عناصر من النوع الصحيح ثم يقوم باستدعاء دالة أخرى تقوم بحساب المتوسط الحسابي لمجموع عناصر المصفوفة كما يستدعي دالة ثالثة مهمتها طباعة عناصر المصفوفة.

```
class Program
{
 static void Main(string[] args)
  {
   int[] m = new int[5];
   m = read();
   double avg1 = average(m);
   print_arr(m, avg1);
   Console.ReadKey();
 }
 public static int[] read()
 {
  int[] arr = new int[5];
  for (int i = 0; i < 5; i++)
  {
   Console.Write("Enter item " + i + "=");
   arr[i] = int.Parse(Console.ReadLine());
  }
        return arr;
 }
 public static double average(int[] arr)
 {
  int sum = 0; double avg = 0;
  for (int i = 0; i < 5; i++)
  sum = sum + arr[i];
  avg = sum / 5;
  return avg;
 }
 public static void print_arr(int[] arr, double cal)
 {
  for (int i = 0; i < 5; i++)
  Console.WriteLine(arr[i] + " ");
  Console.WriteLine();
  Console.WriteLine("avg= " + cal);
  }
}
```

الدالة الرئيسية

دالة القراءة

دالة حساب المتوسط

دالة الطباعة

في هذا المثال احتجنا بالإضافة إلى الدالة الرئيسية إلى تعريف ثلاثة دوال(Methods):

- دالة لقراءة عناصر المصفوفة.
- دالة لحساب المتوسط الحسابي لعناصر المصفوفة.
- دالة لطباعة عناصر المصفوفة وكذلك طباعة المتوسط.

تسلسل تنفيذ البرنامج السابق

يبدأ البرنامج بالسطر رقم 5 والذي تم فيه حجز مواقع العناصر لمصفوفة "m".

السطر رقم 6 والذي تم فيه استدعاء دالة القراءة (read) لينتقل التنفيذ إلى دالة القراءة في السطر رقم 11.

السطر رقم 11 وتم فيه الإعلان عن دالة تحت اسم "read" لا تستقبل أي معاملات وترجع مصفوفة من القيم حيث يبدأ جسم الدالة من السطر رقم 12.

السطر رقم 13 وتم فيه الإعلان عن مصفوفة تحت اسم "arr" داخل الدالة "read" ليتم فيها تخزين القيم داخل الدالة قبل إرسالها إلى الدالة الرئيسية.

الاسطر من 14 إلى 18 تم فيها استخدام حلقة التكرار "for" في إدخال عناصر إلى المصفوفة "arr".

السطر رقم 19 تم فيه إرجاع القيم التي تم تخزينها في المصفوفة "arr" إلى البرنامج الرئيسي (نقطة الاستدعاء) ليتم تخزينها في المصفوفة "m" في البرنامج الرئيسي من خلال جملة التخصيص في السطر رقم 6 لينتقل التنفيذ إلى السطر رقم 7.

السطر رقم 7 تم فيه استدعاء الدالة الخاصة بحساب المتوسط الحسابي لعناصر المصفوفة كما تم فيه إرسال (تمرير) القيم المخزنة في المصفوفة "m" إلى الدالة لحساب متوسط هذه القيم لينتقل التنفيذ بعدها إلى السطر رقم 21.

السطر رقم 21 وتم فيه الإعلان عن دالة تحت اسم "average" تستقبل مصفوفة من المعاملات (القيم في المصفوفة m) وترجع قيمة واحدة من النوع المضاعف والتي تمثل المتوسط الحسابي للمصفوفة حيث يبدأ جسم الدالة من السطر رقم 22.

السطر 23 تم فيه الإعلان عن عدد اثنان من المتغيرات المحلية (خاصة بالدالة) متغير لتخزين مجموع عناصر المصفوفة ومتغير لتخزين متوسط مجموع العناصر.

الاسطر من 24 إلى 25 تم فيها استخدام حلقة التكرار "for" في حساب مجموع عناصر المصفوفة.

السطر 26 تم فيه حساب المتوسط الحسابي لمجموع العناصر وتخزين النتيجة في المتغير "avg".

السطر رقم 27 تم فيه إرجاع القيمة التي تم تخزينها في المتغير "avg" إلى البرنامج الرئيسي (نقطة الاستدعاء) ليتم تخزينها في المتغير "avg1" في البرنامج الرئيسي من خلال جملة التخصيص في السطر رقم 7 لينتقل التنفيذ إلى السطر رقم 8.

السطر رقم 8 تم فيه استدعاء الدالة الخاصة بطباعة عناصر المصفوفة كما تم فيه إرسال القيم المخزنة في المصفوفة "m" وكذلك قيمة المتوسط الحسابي المخزنة في المتغير "avg1" كمعاملات إلى الدالة الخاصة بالطباعة لينتقل التنفيذ بعدها إلى السطر رقم 29.

السطر رقم 29 وتم فيه الإعلان عن دالة تحت اسم "print_arr" والتي تستقبل معاملين (مصفوفة من القيم (القيم في المصفوفة m) و المتوسط الحسابي لعناصر المصفوفة والمخزن في المتغير avg2) ولا ترجع أي قيمة حيث يبدأ جسم الدالة من السطر رقم 30.

السطر رقم 31 ،32 تم فيه استخدام حلقة "for" لطباعة كافة عناصر المصفوفة.

السطر 34 تم فيه طباعة المتوسط الحسابي للمصفوفة.

تذكير : تعتبر عملية تمرير المصفوفات إلى الدوال عملية "تمرير بالعنوان" ولتوضيح ذلك نقدم المثال التالي:

مثال: يقوم البرنامج التالي بتخزين قيم مصفوفة وطباعتها ثم يقوم بإرسال هذه القيم إلى دالة تقوم بضرب كافة عناصر المصفوفة في 2 والعودة للدالة الرئيسية وطباعة المصفوفة مره أخرى ومن خلال تنفيذ البرنامج سنلاحظ أن قيم عناصر المصفوفة قد تغيرات بناءً على العملية التي حدثت في الدالة

وذلك نظراً لأن المصفوفة الفعلية (الموجودة في جملة الاستدعاء) والمصفوفة الشكلية (المعرفة في رأس الدالة) لهما موقع واحد في الذاكرة.

```
class Program
    {
     static void Main(string[] args)
        {
          int[] a = new int[4] ;
          a[0]=4;a[1]=14;a[2]=47;a[3]=12;
          Console.Write("Array Elements before passing: ");
          printArray(a);
          change(a);
          Console.Write("Array Elements after passing : ");
          printArray(a);
          Console.ReadKey();
        } //end of main
   static void change(int[] b)
   {
    for (int j=0; j < 4; j++)
    b[j]*= 2;
   } //end of method change
   static void printArray(int[] c)
  {
   for (int i=0; i < 4; i++)
   Console.Write(c[i] + "\t");
   Console.WriteLine();
  } //end of method print
}
```


ناتج تنفيذ البرنامج

قيم عناصر المصفوفة قبل إرسالها إلى الدالة :

4 14 47 12

قيم عناصر المصفوفة بعد إرسالها إلى الدالة:

8 28 94 24

مثال 2- اكتب برنامج يقوم باستقبال مصفوفة احادية البعد مكونة من 10 عناصر من النوع الصحيح ثم يقوم باستدعاء دالة تقوم بحساب أكبر قيمة في المصفوفة وكذلك دالة أخرى لحساب اصغر قيمة كما يقوم باستدعاء دالة ثالثة تقوم بطباعة اصغر واكبر قيمة.

```
class Program
    {
      static void Main(string[] args)
      {
       int[]  m= new int[10];
       for (int i =0 ;i <10;i++)
       {
          Console.Write ("enter item "+(i+1)+"=");
           m[i]=int.Parse (Console.ReadLine ());
       }
      int max_element= max(m);
      int min_element= min(m);
      print(max_element,min_element);
      Console.ReadKey();
    }
    public static int max(int[] arr1)
    {
        int max=arr1[0];
        for (int i =0 ;i <10;i++)
            if (arr1[i]>max)
                max=arr1[i];
        return max;
    }
    public static int min(int[] arr2)
    {
        int min=arr2[0];
        for (int i =0 ;i <10;i++)
            if (arr2[i]<min)
                min=arr2[i];
        return min;
    }
    public static void print(int mx,int mn)
    {
        Console.WriteLine("max="+mx);
        Console.WriteLine("min="+mn);
    }

}
```

الدوال و المصفوفات ذات البعدين:

الشكل العام لتعريف دالة تستقبل مصفوفة ذات بعدين:

كما ذكرنا سابقاً فإنه لا يوجد هنالك فرق كبير بين تعريف دالة تحتوي على مجموعة معاملات وبين دالة تحتوي على مصفوفة من المعاملات حيث إن الفرق الوحيد يكمن في تغير قائمة المعاملات بمصفوفة من المعاملات اثناء التعريف كما أنه لا يوجد فرق بين استخدام مصفوفة من البارومترات ذات بعد و احد ومصفوفة ذات بعدين.

- في حالة استقبال مصفوفة ذات بعدين و إرجاع قيمة:

```
Access_modifier   returen_type   method_name (array_type [,]   param_array_name)
{
}
```

- في حالة استقبال مصفوفة ذات بعدين وارجاع مصفوفة ذات بعدين:

```
Access_modifier  returen_type[][]  method_name(array_type[,]    param_array_name)
{
}
```

مثال :اكتب شفرة الإعلان عن دالة تقوم باستقبال مصفوفة ذات بعدين 3*3 ثم تقوم بطباعة هذه القيم (لا تعود بأي قيمة).

```
public static void print_arr(int[,] arr)
        {
            for (int i =0 ;i <3;i++)
            {
                for (int j=0;j<3;j++)
               Console.WriteLine(arr[I,j]+" ");
                Console.WriteLine();
            }
        }
```

الشكل العام لاستدعاء دالة وإرسال مصفوفة من البارامترات لها:

في حالة كانت الدالة لا ترجع أي قيمة method_name(array_name)

في حالة كانت الدالة ترجع قيمة variable_name= method_name(array_name)

مثال : اكتب برنامج يقوم بتخزين قيم لعناصر مصفوفة ثنائية البعد 3*3 ثم يقوم باستدعاء دالة تقوم بطباعة هذه القيم.

```
class Program
    {
        static void Main(string[] args)
        {
        int[,] m= new int[3,3];
       m[0,0]=57; m[0,1]=15; m[0,2]=55;
       m[1,0]=35; m[1,1]=77;  m[1,2]=21;
       m[2,0]=15;m[2,1]=55;m[2,2]=13;
       print_arr(m);
       Console.ReadKey ();
       }
        public static void print_arr(int[,] arr)
        {
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                    Console.Write(arr[i,j] + " ");
                Console.WriteLine();
            }
        }

    }
```

مثال 2- اكتب برنامج يقوم باستقبال مصفوفة ذات بعدين مكونة من 10 عناصر 2*5 من النوع الصحيح ثم يقوم باستدعاء دالة تقوم بحساب اكبر قيمة في المصفوفة وكذلك دالة لحساب اصغر قيمة.

```
class Program
    {
     static void Main(string[] args)
     {
      int[,] m = new int[2, 5];
      for (int i = 0; i < 2; i++)
           for (int j = 0; j < 5; j++)
           {
            Console.Write("Enter Element " + "[" +i+","+j+"]=");
            m[i, j] = int.Parse(Console.ReadLine());
           }
            int max_element = max(m);
            int min_element = min(m);
            Console.WriteLine("max=" + max_element);
            Console.WriteLine("min=" + min_element);
            Console.ReadKey();
     }
     public static int max(int[,] arr1)
     {
            int max = arr1[0, 0];
            for (int i = 0; i < 2; i++)
                for (int j = 0; j < 5; j++)
                    if (arr1[i, j] > max)
                        max = arr1[i, j];
            return max;
      }
      public static int min(int[,] arr2)
      {
            int min = arr2[0, 0];
            for (int i = 0; i < 2; i++)
                for (int j = 0; j < 5; j++)
                    if (arr2[i, j] < min)
                        min = arr2[i, j];
            return min;
      }
    }
```

التحميل الزائد للدوال Method Overloading:

تمكننا لغة الجافا من كتابة اكثر من دالة بنفس الاسم في فئة (Class) واحدة وهذا ما يعرف بالتحميل الزائد للدوال أي هو عبارة عن كتابة اكثر من دالة تحمل نفس الاسم في فئة واحدة مع اختلاف توقيع كل دالة (Method signature) لكي يتم التميز بينها حيث يتمثل توقيع الدالة في قائمة البارومترات الموجودة بين قوسي الدالة، ولكي نستطيع تعريف اكثر من دالة بنفس الاسم داخل فئة واحدة فإن هذه الدوال المعرفة يجب أن تختلف في إحدى ثلاثة اشياء :

1- عدد المعاملات Number of parameters:

مثال1:

```
public static int max(int x, int y)
  {
      // Method body
      return 0;
  }

 public static int max(int x, int y, int z)
  {
      // Method body
      return 0;
  }
```

مثال 2:

```
class Program
{
    static void Main(string[] args)
    {
        disp('a');
        disp('a', 10);
        Console.ReadKey();
    }
    static void disp(char c)
    {
        Console.WriteLine(c);
    }
    static void disp(char c, int num)
    {
        Console.WriteLine(c + " " + num);
    }
}
```

نلاحظ أن عدد المعاملات في الدالة الأولى يختلف عن عدد المعاملات في الدالة الثانية مع وجود نفس الاسم لكلا الدالتين.

-2 نوع بيانات المعاملات Data type of parameters:

مثال1:

```
      public static int max(int x, int y)
        {
            // Method body
            return 0;
        }
       public static int max(double  x, double y)
        {
            // Method body
            return 0;
        }
```

مثال2:

```
class Program
    {
        static void Main(string[] args)
        {
            disp('A');
            disp(10);
            Console.ReadKey();
        }
        static void disp(char c)
        {
            Console.WriteLine(c);
        }
        static void disp( int num)
        {
            Console.WriteLine( num);
        }
    }
```

نلاحظ أن نوع بيانات المعاملات في الدالة الأولى يختلف عن نوع بيانات المعاملات في الدالة الثانية مع وجود نفس الاسم لكلا الدالتين.

-3 ترتيب نوع بيانات المعاملات Sequence of Data type of parameters:

مثال1:

```
public static int max(int x, double y, float z)
     {
         // Method body
         return 0;
     }
public static int max(double x, float y, int z)
     {
         // Method body
         return 0;
     }
```

مثال2:

```
class Program
    {
     static void Main(string[] args)
        {
            disp('x', 51);
            disp(52, 'y');
            Console.ReadKey();
        }
  public static  void disp(char c, int num)
  {
  Console.WriteLine("I am the first definition of method disp");
  }
  public static void disp(int num, char c)
  {
  Console.WriteLine("I am the second definition of method disp");
  }
```

نلاحظ أن ترتيب نوع بيانات المعاملات في الدالة الأولى يختلف عن ترتيب نوع بيانات المعاملات في الدالة الثانية مع وجود نفس الاسم لكلا الدالتين.

الخاتمة

في ختام هذا الكتاب نحمد الله حمداً كثيراً يليق بعظمة سلطانه على ما وفقنا إليه من سداد لإتمام صفحات هذا الكتاب، كما اود أن انوه إلى أنه أثناء إعداد هذا الكتاب حاولت قدر الإمكان اختبار كافة البرامج وذلك بتنفيذها على المترجم الخاص #C 2015 وإدخال بيانات فعلية على كافة البرامج للتأكد من عملها بالشكل الصحيح، ولأن هذا العمل من فعل بني البشر فهو قابل للنقد والتعديل والتصحيح ....

وما توفيقي إلا بالله.....

أرحب بملاحظاتكم واستفساراتكم وتعليقاتكم وتعديلاتكم على عناوين البريد الالكتروني التالية:

Salemaldrugi@yahoo.com
Salem.adrugi@Elmergib.edu.ly

تحياتي

مُعد الكتاب: سالم الدروقي